Titel

[Abkürzungsverzeichnis 3](#_Toc102742115)

[RS-485 4](#_Toc102742116)

[Modbus 6](#_Toc102742117)

[Modbus Frame 6](#_Toc102742118)

[Verzeichnisse 9](#_Toc102742119)

[Quellenverzeichnis 9](#_Toc102742120)

[Abbildungsverzeichnis 10](#_Toc102742121)

[Tabellenverzeichnis 11](#_Toc102742122)

## Abkürzungsverzeichnis

PLC: Programmable Logic Controller

TCP: Transmission Control Protocol

IP: Internet Protocol

TCP/IP: Transmission Control Protocol/Internet Protocol (Internet Protocol Suite)

RTU: Remote Terminal Unit

ASCII: American Standard Code For Information Interchange

ADU: Application Data Unit

PDU: Protocol Data Unit

Dec.: Dezimal

Hex.: Hexadezimal

Bin.: Binär

RS: Recommended Standard

TIA/EIA: Telecommunications Industry Association/Electronic Industrie Alliance

ISO/IEC: International Organization for Standardization/International Electrotechnical Commission

PC: Personal Computer

V: Volt

Abb.: Abbildung

bps: Bit pro Sekunde

### RS-485

RS-485 wurde von der Telecommunications Industry Association und der Electronic Industrie Alliance entwickelt und ist auch unter dem Namen TIA/EIA-485 und ISO/IEC 8482.1993 bekannt [1, S. 185].

RS-485 beschreibt ein Protokoll und dessen physikalische Gegebenheiten um zwei oder mehr Computer miteinander zu verbinden, sodass beide Daten miteinander austauschen können. Diese Computer reichen von PCs und Servern zu kleinsten Mikrocontrollerchips. Ein Alltagsbeispiel ist eine simple Wetterstation, deren Wind-, Feuchtigkeits- oder Luftdrucksensoren via RS-485 von einem PC ausgelesen werden. Der PC kontrolliert in diesem Fall die Sensoren und bestimmt, wann, wie oft und welche Daten sie dem PC senden sollen. Der PC ist der Master (Meister) und die Sensoren die Slaves (Sklaven) [1, S. 1–7].

RS-485 benutzt ausgeglichene Übertragungsleitungen, wo jede Verbindung aus zwei Leitern (A und B) besteht und in jeder Leitung jeweils die diametrale Spannung zur anderen Leitung herrscht. Der Empfänger reagiert auf die Spannungsdifferenzen der beiden Leitungen und diese Übertragungsart wird deshalb auch Differentialsignalisierung genannt. Ein großer Vorteil von einem solchen Leiteraufbau ist die starke Reduzierung von Noise, da die Felder der einzelnen Leitungen sich gegenseitig auslöschen [1, S. 186–187].

RS-485 benötigt mindestens eine 1,5V Differenz zwischen den beiden Ausgängen A und B des Senders. Die Differenz zur Erde ist nicht definiert, darf aber nicht mehr als 7 Volt nach oben und nach unten betragen. Bei dem Empfänger muss die Differenz zwischen A und B immer noch mindestens 0,2V betragen um korrekt erkannt zu werden. Wenn A mindestens 0,2V größer als B ist, wird eine 1 registriert und wenn A mindestens 0,2V kleiner als B ist, wird eine 0 registriert. Das heißt, dass auf dem Weg zwischen Sender und Empfänger mindestens 1,3V Noise zu dem Signal hinzukommen können, es aber trotzdem erkannt wird [1, S. 190–191].

Eine RS-485-Verbindung kann entweder bis zu 10 Mbps schnell oder bis zu 1200 Meter lang sein. Wie man in Abb. 1 sieht, sinkt die Übertragungsrate ab 100 Kbps aufgrund der erhöhten Kapazität des Kabels bei langer Länge ab.

Abb. RS-485 Geschwindigkeit in Abhängigkeit der Kabellänge [1, S. 193]

asdasdasfasfas

### Modbus

Modbus ist ein Datenübertragungsprotokoll, welches von Modicon, jetzt Schneider Electric, im Jahr 1979 entwickelt wurde. Es wurde schon von hunderten verschiedenen Herstellern in tausenden verschiedenen Geräten eingesetzt und ist damit der de facto Standard um PLCs, Computer, Sensoren und Aktoren miteinander zu verbinden [2, S. 508], [3]. Es gibt verschiedene Versionen von Modbus, unter anderem Modbus RTU, Modbus ASCII, Modbus über TCP/IP, Modbus Plus und Modbus Enron [2, S. 509], [3]–[5]. Serieller Modbus (Modbus RTU, Modbus ASCII,) beruht auf dem Master/Slave-System, wo eine einziges Master-System bis zu 247 Slave-Systeme nach Daten abfragen kann [2, S. 508]. Die folgenden Kapitel behandeln nur Modbus RTU.

Das Master-System kann zwei Arten von Nachrichten (Requests) an Slave-Systeme schicken: Unicast, wo das Master-System genau ein Slave-System anspricht, welches einen Prozess ausführt und ihm danach mit einer Reply antwortet, und Broadcast, wo das Master-System allen Slave-Systemen eine Request schickt, welche aber nur ein Schreibbefehl sein kann. Bei einem Broadcast wird keine Reply von den Slaves übermittelt.

Der Master sendet Unicast-Nachrichten, auch Requests genannt an jeweils einem Slave. Dieser antwortet dem Master mit einer Response. Eine vollständige Nachricht im Modbus-Protokoll ist in Abb. 2 zu sehen. Diese wird Application Data Unit (ADU) genannt und besteht aus einer Adresse, der Protocol Data Unit (PDU), bestehend aus dem Funktionscode und den zu übertragenden Daten, und schlussendlich dem Fehlercheck.

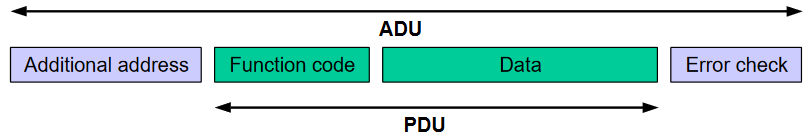
Eine vollständige Nachricht im Modbus-Protokoll ist in Abb. 2 zu sehen. Diese wird Application Data Unit (ADU) genannt und besteht aus einer Adresse, der Protocol Data Unit (PDU), bestehend aus dem Funktionscode und den zu übertragenden Daten, und schlussendlich dem Fehlercheck.   


Abb. 2 Komponenten Modbus Nachricht [6, S. 3]

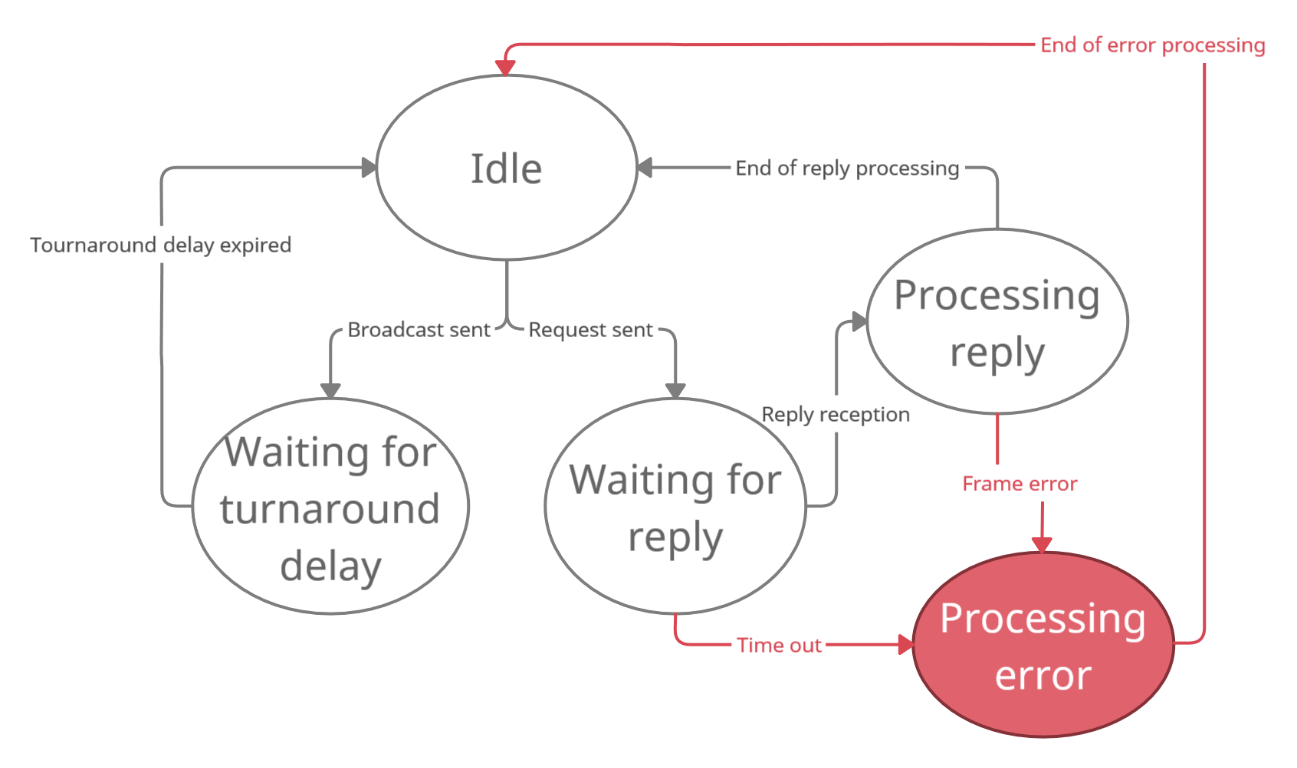


Abb. Modbus Master Zustandsdiagramm [7, S. 9]

In Abb. 3 ist der vereinfachte Prozessablauf eines Modbus-Master-Systems dargestellt. Dieser empfängt von einem im OSI-Modell höher liegenden Protokoll Befehle. Das Master-System ist ohne Befehl von außen im Ruhemodus (Idle). Wenn nun ein Protokoll einen Prozess in einem oder mehreren Slaves ausführen will, sendet es einen Befehl an das Master-System. Wenn es ein Broadcast-Befehl ist, sendet der Master diesen an alle Slaves und wartet dann eine vorher definierte Zeit (Turnaround delay) bevor es wieder in den Ruhemodus zurück geht. Bei einem Unicast-Befehl sendet der Master diesen an das ausgewählte Slave-System und wartet auf eine Reply. Wenn diese nicht nach einer im Voraus festgelegten Zeit zurückkommt oder diese empfangen wurde, aber Fehler beinhaltet, geht der Master zurück in den Ruhemodus. Wenn eine fehlerfreie Reply innerhalb des festgelegten Zeitfenster empfangen wird, wird diese an das auftraggebende Protokoll gegeben [7, S. 9–10].

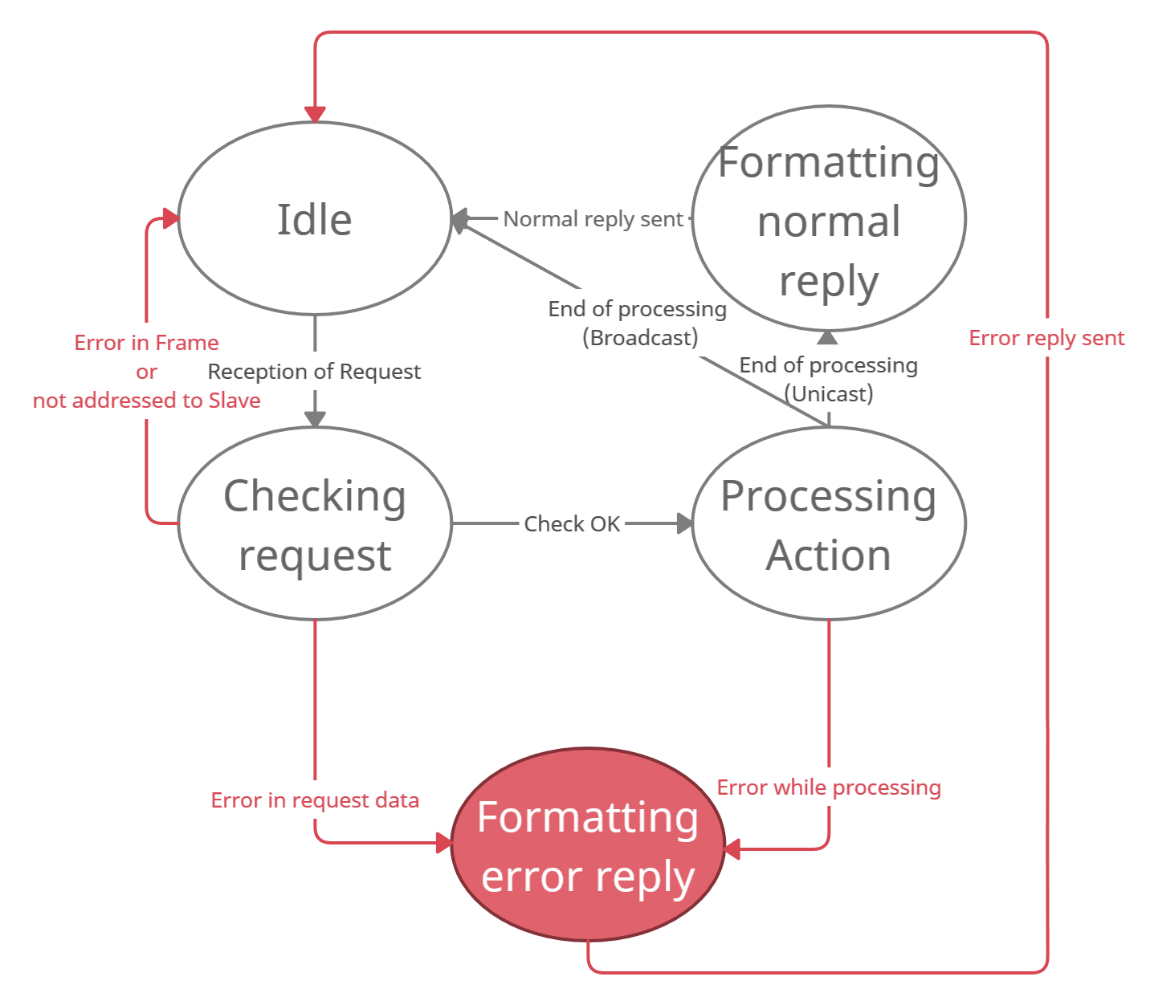


Abb. Modbus Slave Zustandsdiagramm [7, S. 10]

In Abb. 4 ist der komplette Prozessablauf eines Modbus Slaves dargestellt. Beim Starten des Slaves-Systems befindet sich dieses im Ruhemodus (Idle). Wenn der Slave eine Nachricht empfängt, wird diese zuerst auf das korrekte Format überprüft. Außerdem stellt der Slave fest, ob die Nachricht an seine Adresse gerichtet ist. Wenn die Nachricht förmliche Fehler oder die falsche Adresse enthält, wird diese Nachricht ignoriert und der Slave geht zurück in den Ruhemodus. Ist die Nachricht an den Slave adressiert und förmlich korrekt, wird zuerst überprüft ob er den Befehl in dieser Nachricht ausführen kann. Danach wird der Befehl ausgeführt. Wenn bei diesen beiden Prozessen Fehler vorkommen, wird eine Fehlernachricht (error reply) an den Master geschickt, die einem dem Fehler entsprechenden Code enthält. Gibt es keine Fehler, geht der Slave bei einer Broadcast-Nachricht direkt nach der Ausführung, und bei eine Unicast-Nachricht nach dem Senden der Antwort (Reply), in den Ruhemodus [7, S. 10].

Die Modbus Nachricht, auch Frame genannt, startet und endet mit mindestens 3,5 Bitübertragungszeitdauer Stille. Wenn in dem Frame mehr als 1,5 Bitübertragungszeitdauer Stille enthalten sind, wird es vom Empfänger verworfen. Das Adressfeld enthält acht Bit und es gibt 247 valide Adressen. Der Master adressiert den Slave mit seiner Adresse. Ein Slave antwortet dem Master mit seiner eigenen Adresse im Adressfeld. Im Funktionscodefeld steht eine acht Bit lange Nachricht, die dem Slave sagt, was er genau zu machen hat. Es gibt elf verschiedene Funktionscodes, welche alle Modbusübertragungsmedien unterstützen, welche in TABELLE I dargestellt sind und mehrere Dutzende, welche Übertragungsmedium spezifisch sind. Das Datenfeld ist n\*8 Bit lang und enthält die zu übertragenden Daten. Als letztes kommt das 16-bit-lange Fehlercheckfeld, wo mit Hilfe von Cyclical-Redundancy-Check-Rechnungen die empfangene Nachricht überprüft wird [8, S. 18–25].

TABELLE ALLGEMEINE MODBUS FUNKTIONEN [8, S. 31–66]

|  |  |  |
| --- | --- | --- |
| Code in Dec. | Name | Funktion |
| 01 | Read Coil Status | Liest den AN/AUS Status eines Coils |
| 02 | Read Input Status | Liest den AN/AUS Status eines Inputs |
| 03 | Read Holding Register | Liest die Werte von Holding Registern |
| 04 | Read Input Register | Liest die Werte von Input Registern |
| 05 | Force Single Coil | Setzt den AN/AUS Status eines Coils |
| 06 | Preset Single Register | Setzt die Werte von einem Holding Register |
| 07 | Read Exception Status | Liest den Status von prädefinierten Status Coils. Bspw.: Maschine in Betrieb |
| 08 | Diagnostic | Mehrere Unterfunktionen für verschieden Status von Slaves |
| 15 | Force Multiple Coils | Setzt den AN/AUS Status mehrere Coils |
| 16 | Preset Multiple Register | Setzt die Werte von mehreren Holding Register |
| 17 | Report Slave ID | Zeigt die Slave ID an |



Abb. Beispiel Modbus Funktion 03[8, S. 15]

In Abb. 3 kann man eine Beispielsregisterabfrage sehen. Es wurde nur die PDU dargestellt, bei der gesamten Nachricht wurde am Anfang noch die Adresse und am Ende die CRC-Summe übermittelt. Alle Registeradressen und Registerwerte werden in einen High- und Lowteil aufgeteilt, welche hintereinander gesetzt werden müssen. Register Value Hi \* 0x100 + Register Value Lo = Register Value (1). Der Master fragt dort die Daten von Register 0x6B, 0x6C und 0x6D ab, indem das abzufragende Startregister und die Anzahl der darauffolgenden Register definiert werden. Der Slave antwortet daraufhin erst mit der Funktion und den Bitanzahl der Antwortwerte. Danach mit den Werten der abgefragten Register. So weiß der Master, dass Register 0x6B = 0x022B, Register 0x6C = 0x000 und Register 0x6D = 0x0064 ist.

Die Query sagt dem adressierten Slave via dem Funktionscode was er tun soll. Beispielsweise sagt der Code 03, dass der Slave Register auslesen und mit deren Inhalt antworten soll. Im Datenfeld ist dann das Register, welches zuerst ausgelesen werden soll, und die Anzahl der drauffolgenden auszulesenden Register. Der Slave antwortet darauf mit seiner Adresse und dem Funktionscode, welchen er ausgeführt hat. Im Datenfeld sind dann aber die ausgelesenen Registerdaten enthalten [8, S. 14].

# Verzeichnisse

## Quellenverzeichnis

[1] J. Axelson, *Serial Port Complete: Programming and Circuits for RS-232 and RS-485 Links and Networks.* Madison: Lakeview Research, 2002.

[2] B. Drury, *The control techniques drives and controls handbook*, 2nd ed. Stevenage: Institution of Engineering and Technology, 2009.

[3] „Modbus FAQ“. https://modbus.org/faq.php (zugegriffen 5. Mai 2022).

[4] „About Modbus | Simply Modbus Software“. http://www.simplymodbus.ca/FAQ.htm (zugegriffen 5. Mai 2022).

[5] „Modbus Plus | Schneider Electric USA“. https://www.se.com/us/en/product-range/576-modbus-plus/ (zugegriffen 5. Mai 2022).

[6] Modbus Organization, Inc, „MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3“. 26. April 2012. [Online]. Verfügbar unter: https://modbus.org/specs.php

[7] Modbus Organization, Inc, „MODBUS over Serial Line Specification and Implementation Guide V1.02“. 20. Dezember 2006. Zugegriffen: 20. Mai 2022. [Online]. Verfügbar unter: https://modbus.org/docs/Modbus\_over\_serial\_line\_V1\_02.pdf

[8] MODICON, Inc., Industrial Automation Systems, „Modicon Modbus Protocol Reference Guide“. Juni 1996.

## Abbildungsverzeichnis

[Abb. 1 RS-485 Geschwindigkeit in Abhängigkeit der Kabellänge [1, S. 193] 5](#_Toc102742123)

[Abb. 2 Komponenten Modbus Nachricht [6, S. 3] 6](#_Toc102742124)

[Abb. 3 Beispiel Modbus Funktion 03[7, S. 15] 7](#_Toc102742125)

## Tabellenverzeichnis

[TABELLE I ALLGEMEINE MODBUS FUNKTIONEN [7, S. 31–66] 7](#_Toc102742126)